CSN08x14

Scripting for Cybersecurity and Networks

OPTIONAL Lecture 12: Python and databases

In this lecture

- *Python and databases*
  - Focus on SQLite (browser history, satnav)

Interaction with databases

Why interact with databases?

You may think that in Cybersecurity you won't have much reason for querying say an Oracle database, but…

Computers and mobile devices use many databases to store data, which may be important for us,

For example: ….

Examples of databases important in Cybersecurity

- User data:
  - *web browser history,*
  - *satnav records,*
  - *contacts, …*
- System data:
  - *Registered users, when logged on and from where, …*
  - *Error logs*
- Websites may store logs of requests made to the site and their origin
- Wireshark files are effectively a database of network packets

Python and databases

- There are many libraries available for interacting with databases.
- Select the appropriate one for your database management system
- For example, use **sqlite3** library for **SQLite** databases

Why focus on SQLite?

- Because each type of database uses a different Python library, we need to pick one as an example
- SQLite databases are often important in Cybersecurity, particularly digital forensics, because they are used behind the scenes of many apps and devices, including:
  - *Firefox (to store browsing history)*
  - *Chrome*
  - *Safari*
  - *Satnav devices (to store history of places visited)*

Some Users of SQLite

some satnavs

http://www.sqlite.org/famous.html

What is SQLite

- A very "light" open source DBMS (Database Management System)
  - *embedded SQL database engine*
    - no separate server process; reads and writes directly to ordinary disk files; cross-platform (32-bit / 64-bit, big-endian / little-endian architectures).
  - *Very compact – requires little space / RAM*
    - Popular for mobile devices
  - *Popular as an application file format*
    - Used “behind the scenes” of many apps.

http://www.sqlite.org/about.html

SQLite Example 1: Satnav records

Satnav records courtesy of Dr Ian Ferguson, Abertay University.

As these are not my own files, they are not provided to you. However, the screenshots and examples below illustrate how Python could be used to analyse these.

In the lab, you will work with Chrome browsing history, which is briefly introduced at the end of this lecture as Example 2.

Example: Satnav records

- SQLite database with 2 tables:
  - *Item: contains individual place records, stored every 10 seconds*
  - *Log: contains overview of journeys*

Name	Type
Tables (2)	
Item	
itemId	INTEGER
logId	INTEGER
at	DATETIME
latitude	REAL
longitude	REAL
track	REAL
speed	REAL
height	REAL
Log	
logId	INTEGER
routeId	INTEGER
aircraftId	INTEGER
startDate	DATETIME
endDate	DATETIME
distance	INTEGER
startLocation	VARCHAR
endLocation	VARCHAR

DB Browser for SQLite - C:\Users\Petra\Dropbox\CSN08114 Python\satnav\FlightLog.sqlite

File Edit View Help

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Item

New Record | Delete Record

	itemId	logId	at	latitude	longitude	track	speed	height
	Filter	F...	Filter	Filter	Filter	Filter	Filter	Filter
430	433	10	2010-02-08 13:27:41.016	56.4620633333333	-3.40421333333333	250.519	109.8659973144...	39.089
431	432	10	2010-02-08 13:27:31.016	56.4634083333333	-3.39590333333333	254.495	110.4589996337...	62.083
432	431	10	2010-02-08 13:27:21.016	56.4644683333333	-3.38717	258.125	111.2340011596...	79.637
433	430	10	2010-02-08 13:27:11.016	56.46405	-3.37844333333333	261.136	107.2480010986...	380.594
434	429	9	2010-02-0					201.917
435	428	9	2010-02-0					201.917
436	427	9	2010-02-08 13:07:08.015	56.5056566666667	-3.366945	314.489	0.0	201.917
437	426	9	2010-02-08 13:06:58.015	56.5056566666667	-3.366945	314.489	0.0	201.917
438	425	9	2010-02-08 13:06:48.015	56.5056566666667	-3.366945	314.489	0.0	201.917

The Item table contains entries made at 10 second intervals, each giving location and speed

|< < 430 - 438 of 862 > >|

Go to: 1

UTF-8

DB Browser for SQLite - C:\Users\Petra\Dropbox\CSN08114 Python\satnav\FlightLog.sqlite

File Edit View Help

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Log

New Record | Delete Record

	logId	routeId	aircraftId	startDate	endDate	distance	startLocation	endLocation
	Filter	Fil...	Filter	Filter	Filter	Filter	Filter	Filter
1	1	-1	-1	2010-02-04 07:47:42.064	2010-02-04 07:52:12.064	0	-3.47 56.45	
2	2	-1	-1	2010-02-04 14:46:31.031	2010-02-04 14:58:54.029	0	-4.21 55.91	
3	3	-1	-1	2010-02-04 15:00:25.031	2010-02-04 15:40:50.07	0	-3.83 56.04	
4	4	-1	-1	2010-02-04 15:42:00.07	2010-02-04 18:56:03	0	-3.45 56.4	
5	5	-1	-1	[illegible]	[illegible]	[illegible]	[illegible]	
6	6	-1	-1	[illegible]	[illegible]	[illegible]	[illegible]	
7	7	-1	-1	2010-02-05 12:43:33.052	2010-02-05 12:51:13.052	0	-3.47 56.45	
8	8	-1	-1	2010-02-05 12:57:43.052	2010-02-05 13:03:13.052	0	-3.42 56.4	SCONE
9	9	-1	-1	2010-02-05 13:40:54.038	2010-02-08 13:07:28.015	0	-3.37 56.43	

The Log table contains one entry for every journey, each giving start and end date plus start location

|< < 1 - 9 of 11 > >|

Go to: 1

UTF-8

Example continued

To understand the user's movements, we may want to:

- Create a summary of the journeys
  - *For further analysis this could contain e.g. average speed and height as well as start date and duration*
- Plot the individual items on a map (Flythrough?)
  - *Need to extract appropriate info from DB*
  - *Write KML (see lecture 11)*

Steps for querying a database with Python

1. Import the appropriate Python library (e.g. sqlite3)
2. Establish a connection to the database by creating a Connection object.
3. Create a Cursor object using the cursor method of the Connection object.
4. Execute a SELECT query
   - *Using SQL appropriate for the database (you need to know the db!)*
5. Call the fetchall() method of the cursor object to fetch the data
   - *Result is a list of tuples.*
   - *Process further if required using standard Python methods*

6. Repeat steps 3-5 for each query you want to run

Adapted from: http://www.sqlitetutorial.net/sqlite-python/sqlite-python-select/

DB query Example: Steps 1&2

1. Import the appropriate Python library (e.g. sqlite3)

2. Establish a connection to the database by creating a Connection object.

```python
# modified from
# http://www.sqlitetutorial.net/sqlite-python/sqlite-python-select/
# Nov 2018 PEP8
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file"""
    try:
        conn = sqlite3.connect(db_file)
        return conn  # if connection works
    except Error as e:
        print(e)
    return None  # if connection fails
```

DB query Example: Steps 3-5 (first query)

3. Create a Cursor object using the cursor method of the Connection object.

4. Execute a SELECT query
   - *Using SQL appropriate for the database (you need to know the db!)*

5. Call the fetchall() method of the cursor object to fetch the data
   - *Result is a list of tuples.*

```
def selected_locations(conn, startID, n=10):
    """id, timestamp, longitude, latitude, altitude of 10 items"""
    cur = conn.cursor()
    query = f"SELECT itemID, longitude, latitude, height \
              FROM Item \
              WHERE itemID between {startID} and {startID+n}"
    cur.execute(query)
    rows = cur.fetchall()
    return(rows)
```

DB query Example: Steps 3-5 (second query)

3. Create a Cursor object using the cursor method of the Connection object.

4. Execute a SELECT query
   - *Using SQL appropriate for the database (you need to know the db!)*

5. Call the fetchall() method of the cursor object to fetch the data
   - *Result is a list of tuples.*

```
def all_journey_stats(conn):
    """journeys and their duration in minutes"""
    cur = conn.cursor()
    query = "SELECT logID, \
            (strftime('%s',t.endDate)-strftime('%s',t.startDate))/60 \
            AS duration_mins \
            FROM Log AS t;"
    cur.execute(query)
    rows = cur.fetchall()
    return(rows)
```

DB query Example: output

- This just prints the lists of tuples as they are
- Could process this further.
- For example, use list of locations as input for converting to KML

```
============ RESTART: F:\Dropbox\CSN08114 Python\satnav\satnav.py ============
Selected locations (ID,logitude,latitude,height):
[(430, -3.3784433333333337, 56.46405, 380.594), (431, -3.38717, 56.4644683333333
36, 79.637), (432, -3.3959033333333335, 56.463408333333334, 62.083), (433, -3.40
42133333333333, 56.46206333333333, 39.089), (434, -3.412791666666667, 56.4608033
3333333, 14.778), (435, -3.4218683333333333, 56.45886, 80.47)]

Journey IDs and duration in minutes:
[(1, 4), (2, 12), (3, 40), (4, 194), (5, 7), (6, 6), (7, 7), (8, 5), (9, 4286),
(10, 21), (11, 51)]
```

Structure | Browse & Search | Execute SQL | DB Settings

TABLE Item | Search | Show All | Add | Duplicate | Edit | Delete

itemId	logId	at	latitude	longitude	track	speed	height
1	1	2010-02-04 07:47:42.064	56.44698	-3.471805	177.391	30.3199996...	33.101
2	1	2010-02-04 07:47:52.064	56.44518	-3.4716683333...	178.182	41.9510002...	27.619
3	1	2010-02-04 07:48:02.064	56.44319	-3.4717033333...	181.882	41.6489982...	29.596
4	1	2010-02-04 07:48:12.064	56.44129	-3.4720016666...	186.483	42.3180007...	33.081
5	1	2010-02-04 07:48:22.064	56.4393	-3.4724666666...	187.271	42.9500007...	32.641
6	1	2010-02-04 07:48:32.064	56.437338333...	-3.4729133333...	186.952	42.9650001...	27.626
7	1	2010-02-04 07:48:42.064	56.435336666...	-3.473			
8	1	2010-02-04 07:48:52.064	56.433303333...	-3.472			
9	1	2010-02-04 07:49:02.064	56.431388333...	-3.471			
10	1	2010-02-04 07:49:12.064	56.429658333...	-3.469			
11	1	2010-02-04 07:49:22.064	56.427835	-3.468			
12	1	2010-02-04 07:49:32.064	56.4259	-3.467			
13	1	2010-02-04 07:49:42.064	56.423983333...	-3.467075	170.056	41.1829986...	10.904
14	1	2010-02-04 07:49:52.064	56.422183333...	-3.46651	169.704	37.9029998...	9.623

<< | < | 1 to 100 of 862 | > | >>

SQLite database with GPS data (e.g. from SatNav)

Read database table / extract data with Python **sqlite3** module

convert to KML with Python **simplekml** module

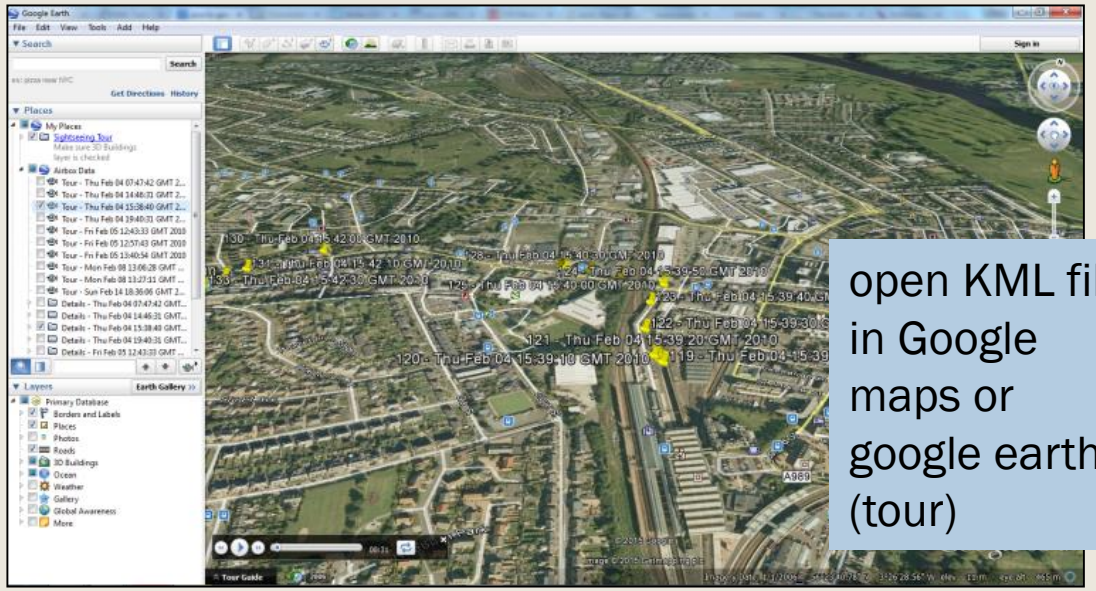
test1.kml

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2">
    <Document>
    <gx:Tour>
        <name>Thu Feb 04 07:47:42 GMT 2010</name>
        <gx:Playlist>
            <gx:FlyTo>
                <gx:duration>5.0</gx:duration>
                <gx:flyToMode>smooth</gx:flyToMod
                <LookAt>
                    <longitude>-3.471805</longit
                    <latitude>56.44698</latitude
                    <altitude>33.101</altitude>
                    <heading>177.391</heading>
                    <range> 1000</range>
                    <tilt>63</tilt>
                </LookAt>
            </gx:FlyTo>

            <gx:FlyTo>
                <gx:duration>5.0</gx:duration>
                <gx:flyToMode>smooth</gx:flyToMode>
                <LookAt>
                    <longitude>-3.4716683333333336</longitude>
                    <latitude>56.44518</latitude>
                    <altitude>27.619</altitude>
                    <heading>178.182</heading>
                    <range> 1000</range>
                    <tilt>63</tilt>
                </LookAt>
            </gx:FlyTo>

            <gx:FlyTo>
                <gx:duration>5.0</gx:duration>
                <gx:flyToMode>smooth</gx:flyToMode>
                <LookAt>
```

kml file suitable for google earth/ google maps



open KML file in Google maps or google earth (tour)

SQLite Example 2: Chrome Browser History

Chrome SQLite files

- Google Chrome stores the entire browsing history of a user
- Many separate files
  - *Many of the key files are SQLite databases*
  - *Windows 10 location: C:\Users\%USERNAME%\AppData\Local\Google\Chrome\User Data\Default*
- see http://www.forensicswiki.org/wiki/Google_Chrome
- Firefox and Safari use similar ideas, but the structure of the directory and the databases are different.

Chrome (v.64, Mar 2018)

- Application Cache
- blob_storage
- Cache
- data_reduction_proxy_leveldb
- databases
- Download Service
- Extension Rules
- Extension State
- Extensions
- Feature Engagement Tracker
- File System
- GCM Store
- GPUCache
- IndexedDB
- JumpListIconsMostVisited
- JumpListIconsRecentClosed
- Local App Settings
- Local Extension Settings
- Local Storage
- Managed Extension Settings
- Media Cache
- Pepper Data
- Platform Notifications
- Service Worker
- Session Storage
- Storage
- Sync Data
- Sync Extension Settings
- Thumbnails
- VideoDecodeStats
- Web Applications

- Affiliation Database
- Affiliation Database-journal
- Bookmarks
- Bookmarks.bak
- Cookies
- Cookies-journal
- Current Session
- Current Tabs
- Custom Dictionary.txt
- Custom Dictionary.txt.backup
- DownloadMetadata
- Extension Cookies
- Extension Cookies-journal
- Favicons
- Favicons-journal
- Google Profile.ico
- History
- History-journal
- Last Session
- Last Tabs

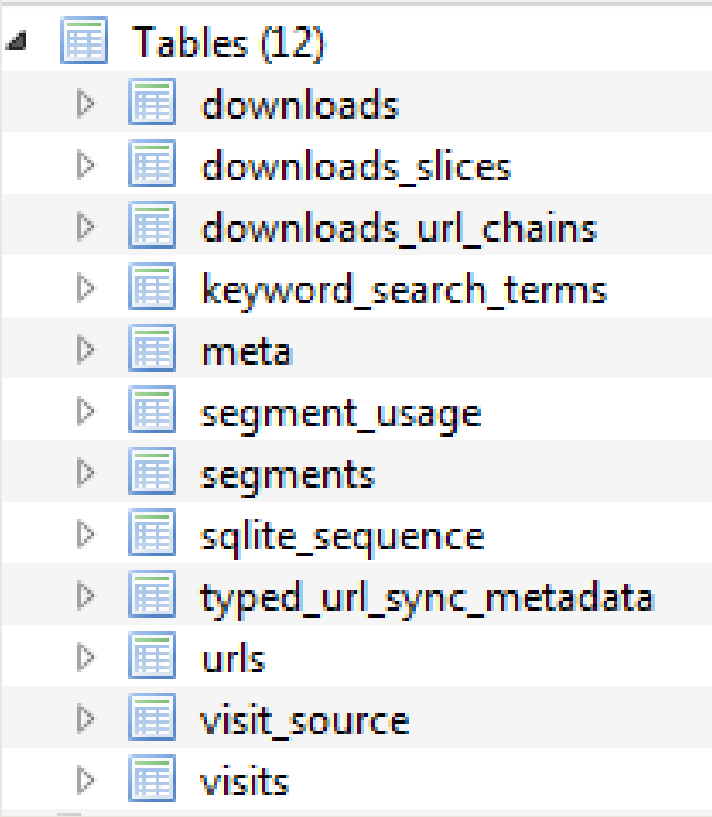
- Login Data
- Login Data-journal
- Network Action Predictor
- Network Action Predictor-journal
- Network Persistent State
- Origin Bound Certs
- Origin Bound Certs-journal
- Preferences
- previews_opt_out.db
- previews_opt_out.db-journal
- QuotaManager
- QuotaManager-journal
- Secure Preferences
- Shortcuts
- Shortcuts-journal
- Top Sites
- Top Sites-journal
- Translate Ranker Model
- TransportSecurity
- Visited Links
- Web Data
- Web Data-journal

Forensically Most useful SQLite files highlighted yellow

Sqlite yellow (no fill where not much use); JSON green; SNSS (session saver) files purple

Chrome – History tables

- Browsing history is stored under the Default folder as "History" and can be examined using any SQLite browser. The available tables (v.64) are:



history example (urls table)

Table: urls

New Record | Delete Reco

	id	url	title	visit_count	typed_count	last_visit_time	hidden	favicon_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	109	http://www.modules.napier.ac.uk/	Modules Information	2	0	13114003955629840	0	0
2	116	https://tracker.napier.ac.uk/	Tracker: Student Management	5	0	13116947102185278	0	0
3	191	https://hrconnect.napier.ac.uk/mthrpr...		10	0	13114606794322029	1	0
4	192	https://hrconnect.napier.ac.uk/mthrpr...		10	0	13114606794322029	1	0
5	273	http://www.ee.surrey.ac.uk/Teaching...	UNIX / Linux Tutorial for Beginners	0	0	0	0	0
6	286	http://archive.oreilly.com/linux/cmd/	Linux Command Directory: Index	0	0	0	0	0
7	299	http://www.nationwide.co.uk/	Nationwide Building Society | On your side	6	5	13116760498332133	0	0
8	315	https://www.facebook.com/	Facebook	67	0	13116369720369144	0	0
9	316	https://www.linkedin.com/	Welcome! | LinkedIn	6	2	13114632954214338	0	0
10	327	http://www.facebook.com/	(13) Facebook	23	0	13116354070834011	0	0
11	330	https://www.evernote.com/Home.action	Evernote Web	5	0	13114708051321772	0	0
12	341	https://evernote.com/	The note-taking space for your life's work | Ev...	3	0	13114708042925936	0	0
13	342	https://www.evernote.com/	The note-taking space for your life's work | Ev...	3	0	13114708042925936	0	0
14	344	http://k2b-bulk.ebay.co.uk/ws/eBayIS...	Sign in or Register | eBay	0	0	0	0	0
15	345	http://www.davidlloyd.co.uk/home	Gym Membership, Racquets, Classes & Swimm...	0	0	0	0	0

(Chrome uses WebKit timestamps – add a 0 at the end to convert to LDAP/NTFS time)

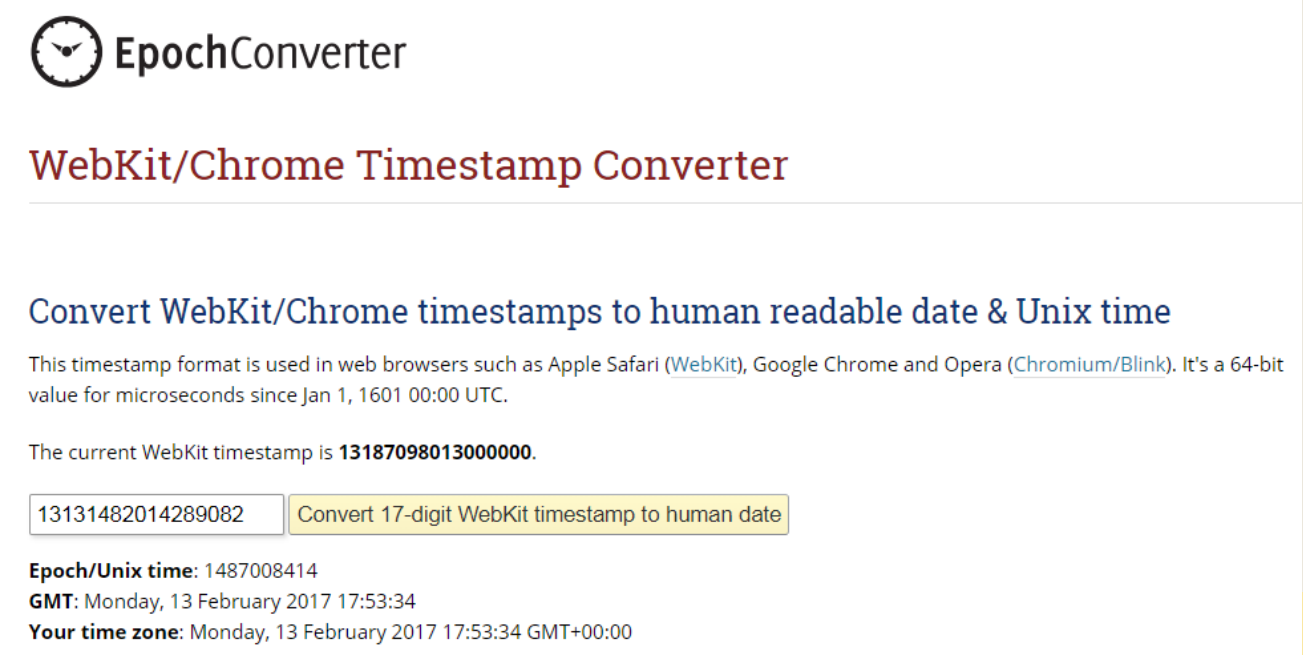
history example (downloads table (extract))

- Example:

	id	current_path	start_time	received_bytes	opened	interrupt_reason	last_modified	mime_type	tab_url	site_url
1	775	C:\Users\40009856\Desktop\DB.Browser.for.SQLite-3....	13116940931338290	13683112	1	0	Wed, 24 Aug 2016...	application/octet-stream	https://github.com/sqlitebrows...	https://google.co.uk/
2	776	C:\Users\40009856\Desktop\fiddlersetup.exe	13116941723912831	2981160	0	0	Tue, 26 Jul 2016 1...	application/octet-stream	https://www.telerik.com/downl...	https://telerik.com/
3	771	C:\Users\40009856\Downloads\CIATA5 06161.pdf	13116448475856546	109253	1	0	Tue, 07 Jun 2016 ...	application/pdf	http://www.principality.co.uk/S...	http://moneysavingexpert.com/
4	782	C:\Users\40009856\Desktop\Forensic_Investigation_of...	13117028777231653	2012255	1	0	Fri, 02 Oct 2015 0...	application/pdf	https://www.academia.edu/163...	https://google.co.uk/
5	649	C:\Users\40009856\Downloads\A29wp 223 IOT.pdf	13110545936439963	617661	1	0		application/pdf		
6	755	E:\masters projects\2005_hall_referencing.pdf	13114603545086081	45258	1	0	Thu, 28 Feb 2013 ...	application/pdf	http://moodle.napier.ac.uk/plu...	http://napier.ac.uk/
7	783	E:\CSN09101 Networked Services\backup-CSN09101_...	13117031656204965	7559029	0	0	Tue, 30 Aug 2016 ...	application/vnd.moodle.backup	http://moodle.napier.ac.uk/bac...	http://napier.ac.uk/
8	784	E:\CSN09101 Networked Services\week2.ppt	13117032517743834	1146368	1	0	Fri, 12 Sep 2014 1...	application/vnd.ms-powerpoint	http://moodle.napier.ac.uk/mo...	http://napier.ac.uk/
9	785		13117040141001822	0	0	40	Tue, 30 Aug 2016 ...	application/vnd.openxmlformats-officedocument.pr...	http://moodle.napier.ac.uk/mo...	http://napier.ac.uk/
10	120	C:\Users\40009856\Downloads\title_list.xlsx	13091819458603097	12422602	1	0	Tue, 07 Jul 2015 1...	application/vnd.openxmlformats-officedocument.spr...		
11	651	E:\masters projects\ProjectDiary.docx	13111505495975157	12932	1	0	Wed, 25 May 201...	application/vnd.openxmlformats-officedocument.wo...	http://moodle.napier.ac.uk/cou...	http://napier.ac.uk/
12	774	C:\Users\40009856\Desktop\free-sqlite-viewer.exe	13116940294031512	13954192	1	0	Mon, 31 Aug 201...	application/x-msdownload	http://www.sqliteviewer.org/dat...	https://google.co.uk/
13	773	C:\Users\40009856\Desktop\sqlitestudio-3.1.0.zip	13116940013967691	16164724	1	0	Fri, 10 Jun 2016 21...	application/zip	http://sqlitestudio.pl/	https://google.co.uk/
14	718	C:\Users\40009856\Desktop\diffpdf-2.1.3-win32-stati...	13113583882657331	5789531	0	0	Mon, 08 Dec 2014...	application/zip	http://download.cnet.com/Diff...	https://google.co.uk/
15	529	E:\Petra\Dropbox\CSN08705_08111\stego\1984-stega...	13104836507346908	229118	0	0	Sun, 15 Apr 2012 ...	image/png		
16	694		13113394054621961	0	0	40	Mon, 04 Jul 2016 ...	text/html	https://hrconnect.napier.ac.uk/...	https://napier.ac.uk/
17	353	E:\Petra\Dropbox\CSN08705_08111\Lecture 3d editin...	13097706801371129	34637628	0	0	Mon, 22 Oct 2012...	video/mp4		
18	332	C:\Users\40009856\Downloads\Rendered - ffb2086b-...	13097670802371272	44073240	1	0		video/x-msvideo		

Google Chrome timestamps

- Chrome stores the timestamps in "Webkit" format
- The number of microseconds since 1/1/1601.
- Example: 13131482014289082
  - *Converts to 13/2/2017 17:53:34 GMT*
- Online converter at https://www.epochconverter.com/webkit



Google Chrome timestamps

- The Online converter at https://www.epochconverter.com/webkit also gives Python code for conversion:

Programming routines

Python

```
import datetime
def date_from_webkit(webkit_timestamp):
    epoch_start = datetime.datetime(1601,1,1)
    delta = datetime.timedelta(microseconds=int(webkit_timestamp))
    print epoch_start + delta

inTime = int(raw_input('Enter a Webkit timestamp to convert:'))
date_from_webkit(inTime)
```

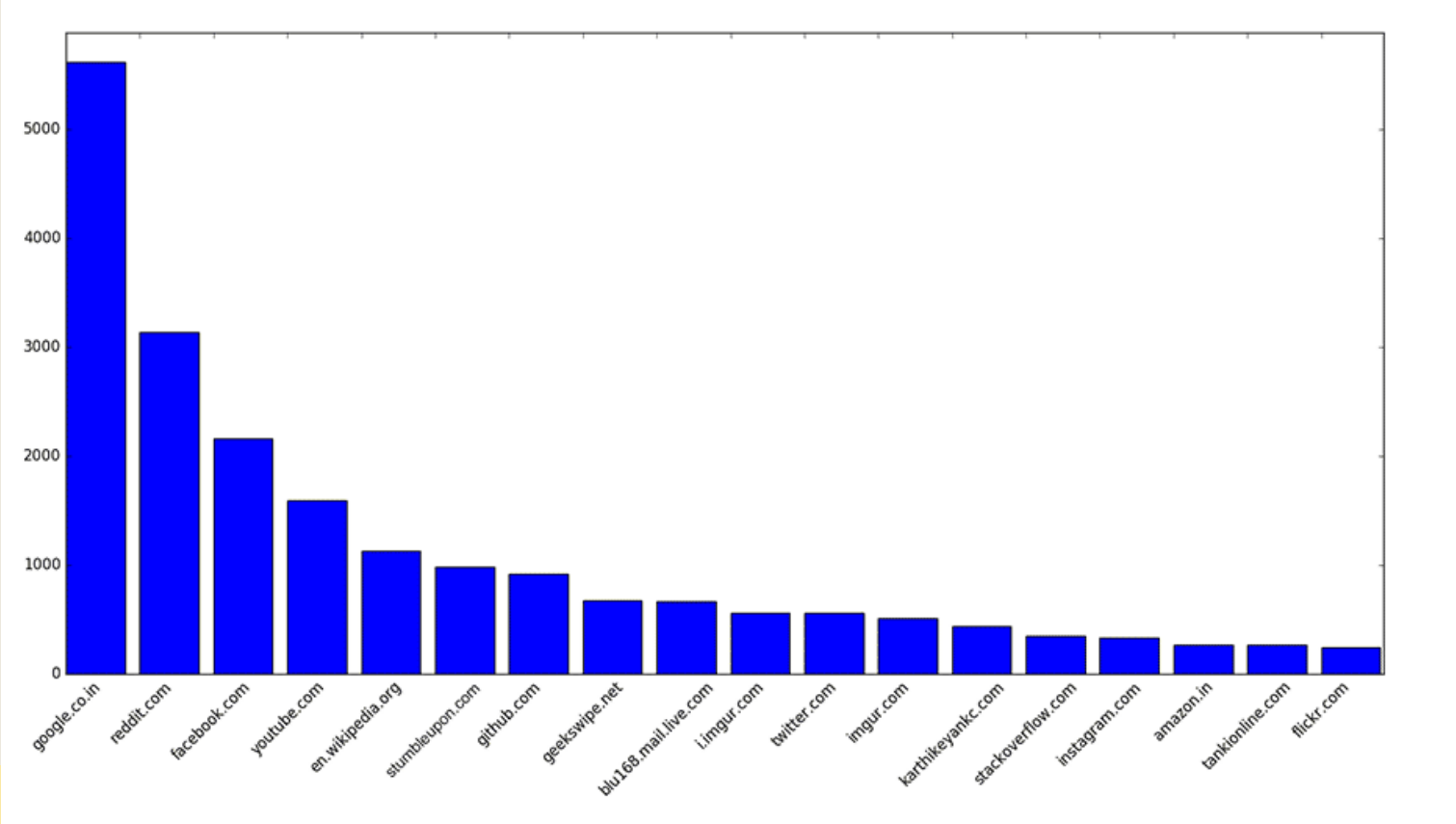
- Conversion can be done directly in an SQL query, e.g.

```
SELECT datetime((last_visit_time/1000000)-11644473600, 'unixepoch','localtime')
AS Decoded_Visit_time FROM urls
```

https://www.forensicfocus.com/Forums/viewtopic/t=12232/

Example Chrome browsing history summary

- Karthikeyan KC provides code to summarise the websites visited at https://geekswipe.net/technology/computing/analyze-chromes-browsing-history-with-python/.



Resources

- Python docs https://docs.python.org/3/library/sqlite3.html
- Excellent article introducing the sqlite3 standard library module: http://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html
- Introduction to SQLite in Python (focus on inserting etc) https://www.pythoncentral.io/introduction-to-sqlite-in-python/
- Example Chrome history analysis with Python https://geekswipe.net/technology/computing/analyze-chromes-browsing-history-with-python/